MDDE: Una concepción genérica para diseño de entornos de desarrollo de software basados en MDSE

César Cuevas, Patricia López Martínez y José M. Drake

Grupo de Ingeniería Software y Tiempo Real, Universidad de Cantabria

{cuevasce,lopezpa,drakej}@unican.es

Resumen. Se presenta ***MDDE*** (*Model-Driven Development Environment*), una concepción genérica para diseño de entornos de desarrollo de software basados en MDSE. Su objetivo es facilitar el uso de esta disciplina a los ingenieros software que diseñan e implementan entornos de soporte a las metodologías que proponen y que necesitan incluir en ellos nuevos modelos de información, herramientas y procesos de desarrollo. El componente principal de la concepción propuesta es un modelo de referencia que define las capacidades básicas, tanto funcionales como de interacción, que son comunes a cualquier entorno. En ella, la especificación e implementación de entornos, el soporte a los procesos que determinan su funcionalidad y la definición de las opciones de interacción, supervisión y control por parte de los operadores, se realizan íntegramente mediante la formulación de modelos. Para dar soporte a esta capacidad, el modelo de referencia incluye un metamodelo que formaliza tales modelos.

Palabras clave: MDSE, meta-modelado, entorno de desarrollo.

1 Introducción

Se presenta *MDDE* (*Model-Driven Development Environment*), una concepción genérica para diseño de entornos de desarrollo de software basados en MDSE. Su objetivo es fomentar la aceptación, adopción y consolidación de MDSE entre los ingenieros software encargados del diseño e implementación de nuevos entornos de soporte a las metodologías que proponen, a los que añaden nuevos modelos de información, herramientas y/o procesos de desarrollo.

El principal componente de *MDDE* es un modelo de referencia (*MDDE framework*) que define las capacidades básicas, tanto funcionales como de interacción, que son comunes a cualquier entorno. Representa una abstracción que especifica sus elementos constituyentes, tanto conceptuales (caracterizados por el papel que juegan) como funcionales (caracterizados por los servicios que prestan y la interfaz que ofrecen), define la arquitectura con que se integran y los mecanismos de interacción entre ellos. Constituye un mapa conceptual independiente de cualquier implementación que proporciona al diseñador de sistemas software la información necesaria para trabajar con un entorno *MDDE* y al diseñador de entornos la vista arquitectural que dirige los elementos que

diseña. Aunque formulado de forma agnóstica respecto a plataforma, para su validación requiere ser implementado sobre una plataforma concreta. Así, dada una plataforma X, la implementación de *MDDE* sobre X se denomina *MDDE-X*, lo cual es ortogonal al dominio y/o metodología a que se pretenda dar soporte. Por ejemplo, podría hablarse de una implementación de *MDDE* sobre Eclipse y concebida para dar soporte al desarrollo de Sistemas de Tiempo Real Embebidos (STRE) basado en MAST [1], con lo cual se trataría del entorno *MDDE-Eclipse-MAST*.

El modelo de referencia puede considerarse desde tres puntos de vista: i) El *punto de vista estructural* contempla los elementos conceptuales que lo constituyen, las relaciones entre ellos y su formalización como metamodelo; ii) el *punto de vista funcional* lo aborda definiendo los elementos que constituyen el motor interno del entorno y las interfaces que utiliza; y iii) *el punto de vista de la implementación*. En esta comunicación se describen principalmente los aspectos estructurales y, como prueba de concepto, también se describe de forma simplificada el ejemplo de implementación *MDDE-MinimalMAST2*. Por motivos de espacio, la exposición del punto de vista funcional queda omitida, aunque en [2] puede encontrarse su estudio en profundidad, incluyendo la idoneidad de Eclipse como plataforma de soporte funcional.

El resto del artículo tiene la siguiente estructura. Tras esta sección introductoria, la sección 2 presenta una visión general de la concepción *MDDE*, mientras que las siguientes secciones abordan el modelo de referencia de *MDDE* desde el punto de vista estructural (sección 3) y de implementación (sección 4). La sección 5 analiza las posibilidades de *MDDE* para el diseñador de entornos. Finalmente la sección 6 aborda trabajos relacionados existentes en la bibliografía y la sección 7 esboza algunas conclusiones y líneas de trabajo futuras.

2 Visión general de *MDDE* y caracterización de los entornos

2.1 Objetivos

Para el diseño de entornos de desarrollo de software bajo *MDDE* se requiere definir las funcionalidades básicas que éstos han de proporcionar: Servir de repositorio (persistente o temporal) de los modelos que contienen la información (introducida o resultante) relativa a los procesos de desarrollo de software, dar soporte a las herramientas con las que se maneja tal información y proporcionar recursos de interacción para que el desarrollador pueda controlar la ejecución de dichos procesos de desarrollo. Asimismo, en la concepción *MDDE* se han considerado otros objetivos, como integrar de forma natural los múltiples aspectos y dominios que conciernen al diseño de un sistema software, proporcionar una estrategia estandarizada e independiente del dominio para acceso a la información y a los procesos, y facilitar el futuro mantenimiento, extensión y portabilidad de los entornos a través de la modularidad, estandarización y reutilización de sus elementos. Todo ello, bajo la consideración de que el entorno de desarrollo va a ser diseñado e implementado por ingenieros software expertos en muy diferentes dominios pero que habitualmente no son expertos en tecnologías MDSE.

2.2 Tipos de entornos *MDDE*

La concepción *MDDE* es única, pero el contenido nativo con que se distribuye un entorno *MDDE* determina su propósito así como su capacidad de evolución y enriquecimiento futuro. Se distingue entre *entorno MDDE especializado*, *extensible* y de *propósito general*. Cada entorno *especializado* tiene por objetivo dar soporte, a través del contenido nativo con que es distribuido, a uno o varios dominios en el desarrollo de software. Sin embargo, los de este tipo no están dotados de los recursos para incorporar nuevos metamodelos y herramientas, por lo que no puede ampliarse su funcionalidad. En cambio, los entornos *extensibles* se distribuyen con recursos nativos que permiten crear nuevos entornos especializados, con los elementos funcionales considerados apropiados para los usuarios finales. Por último, los entornos *de propósito general* se distribuyen dotados tanto de contenido nativo de soporte al desarrollo de software en un cierto conjunto de dominios como de los recursos necesarios para que expertos en nuevos dominios puedan crear nuevos metamodelos y herramientas de soporte a nuevos procesos de desarrollo, extendiendo así persistentemente la funcionalidad del entorno a los nuevos dominios.

2.3 Fundamento operacional

MDDE considera que el operador que usa un entorno para desarrollar proyectos software ejecuta, de forma supervisada, un cierto conjunto de *procesos*, cada uno de los cuales consiste a su vez en la ejecución secuencial o iterativa de operaciones más básicas denominadas *tareas*. De acuerdo al espíritu MDSE, los procesos se formulan como modelos que describen la secuencia de tareas constituyentes y especifican su naturaleza, siendo el entorno el encargado de interpretarlos y ejecutarlos, siempre bajo la supervisión del operador. A su vez, las tareas se formulan como modelos que describen su naturaleza, los modelos sobre los que operan, las transformaciones involucradas, la información ofrecida al operador para la toma de decisiones y las opciones de control con las que supervisa, valida y dirige su ejecución. Así, la especificación y diseño de un entorno va a consistir básicamente en la elaboración de modelos, y no en el desarrollo de su código de implementación. Para dar soporte a esta capacidad, el modelo de referencia incluye un metamodelo que formaliza tales modelos. Como prueba de concepto se ha diseñado e implementado una herramienta que los interpreta, y en base a ellos, genera automáticamente los recursos del entorno diseñado.

3 Modelo de referencia: elementos conceptuales y estructura

Desde un punto de vista estructural, el modelo de referencia de *MDDE* puede describirse en dos niveles. En primer lugar (subsección 3.1), descripción de sus elementos conceptuales, en base a sus atributos e interrelaciones, especificando para cada uno de ellos sus tipos derivados y su clasificación. En segundo lugar (subsección 3.2), la formulación del metamodelo que formaliza los aspectos estructurales del entorno conceptual y sirve de base para la descripción de cada entorno concreto.

3.1 Elementos conceptuales del *MDDE framework*

Modelos. La información manejada en un entorno *MDDE* está formulada como modelos conformes a metamodelos existentes en él. Según su función, se distingue entre modelos de: i) *dominio*, que contienen información sobre un aspecto de un sistema bajo desarrollo (SBD); ii) *herramienta*, que describen la operatividad, configuración o estatus de una herramienta en el entorno y iii) *interacción*, para describir la información que intercambian operador y entorno. Por último, también son modelos los *metamodelos* encargados de describir la información que contendrán los otros modelos.

Atendiendo al ciclo de vida en el entorno, se distingue entre modelos: i) *nativos*, incorporados al entorno durante su creación y por tanto distribuidos de inicio con él; ii) *de proyecto*, con información relevante para el proyecto en desarrollo y producidos bajo demanda del operador o por requerirlo la estrategia de desarrollo y iii) *temporales*, creados transitoriamente para manejar información interna dentro de un proceso del entorno. El conjunto de modelos nativos (dado su carácter, no modificables y no eliminables, aunque pueden ser supervisados o copiados por el operador), junto al resto de elementos nativos, constituye la base de la capacidad funcional y evolutiva de un entorno *MDDE*. En cuanto a los modelos de proyecto, pueden ser construidos manualmente o generados mediante la invocación de procesos en el entorno, y siempre son persistidos en su espacio de datos. El operador o los procesos pueden modificarlos o eliminarlos, pero si esto no ocurre, el entorno los salva persistentemente cuando se cierra el proyecto y los recupera cuando se abre de nuevo. Por último, si la funcionalidad del proceso que genera modelos temporales lo permite, pueden ser supervisados y modificados por el operador en aquellas fases de su ciclo de vida en que son accesibles para él, pero siempre son destruidos cuando finaliza el proceso.

Herramientas *MDDE* (*mTool*). Una *mTool* es un proceso u operación de alto nivel ofrecido por un entorno *MDDE* para algún objetivo de desarrollo o de gestión propia, que enriquece, supervisa o procesa la información contenida en él, y que es invocada explícitamente por el operador desde el propio entorno. Toda *mTool* se compone de una secuencia de actividades más elementales (*tareas*) definidas de forma individual para facilitar su reutilización en otras *mTools*.

Las *mTools* permiten al operador incorporar al entorno, de forma asistida, información relativa al SBD, introduciendo nuevos modelos que enriquecen la información ya existente y presentar la información del SBD disponible en el entorno de acuerdo con el punto de vista que incorpora la *mTool* utilizada, así como procesarla de acuerdo con estrategias de transformación, integración o análisis incorporadas por ella. Permiten además gestionar, también de forma asistida, los modelos del entorno, organizando, persistiendo o eliminando los elementos dentro de él o importar y exportar la información del SBD desde o hacia otros formatos diferentes a los del entorno.

Toda *mTool* se formula a través de un modelo, por lo que su incorporación a un entorno consiste en registrar dicho modelo. Algunas *mTools* son nativas, esto es, sus modelos se han registrado en la creación del entorno y permanecen inalteradas durante todo su ciclo de vida, proporcionándole una funcionalidad operativa básica. Por otro lado, el operador (en el rol de diseñador de entornos) puede definir nuevas *mTools* aportando los correspondientes modelos descriptivos o modificando los de otras ya

existentes. El modelo de una *mTool* describe su: i) funcionalidad, es decir, la secuencia de *tareas* de que se compone; ii) contexto de invocación, esto es, el elemento del entorno desde donde puede ser invocada, bien directamente (menú o botón) o bien por selección contextual; iii) opciones de configuración y iv) información de estado que ha de ir generando mientras se está ejecutando. Los parámetros de configuración están definidos como un submodelo que especifica su identificación, su tipo, valor que se les asigna (siempre tienen al menos valores por defecto) y si pueden ser modificados por el operador. Por su parte, el estado de ejecución se describe también como un submodelo cuyos valores son establecidos de acuerdo con los resultados que se obtienen en fase de ejecución, proporcionando información relevante respecto al éxito/fracaso de la ejecución de las *tareas* internas y respecto a los modelos que va generando la ejecución de la *mTool* en cuestión.

La ejecución de una *mTool* se realiza siempre bajo control y supervisión del operador. Tras la invocación, hay al menos tres puntos en los que se requiere su intervención: 1) Establecer y/o aceptar la configuración de lanzamiento; 2) lanzar la ejecución y 3) dar por concluida la ejecución (supone la eliminación de toda la información de estatus y modelos transitorios generados). Cuando éstos son los únicos puntos de intervención, se trata de ejecución en *modo continuo*, en contraposición a modo *paso a paso*, en que la ejecución requiere que el operador intervenga en la ejecución de cada *tarea* integrante. Como se verá posteriormente en esta misma subsección, para gestionar la ejecución de una *mTool*, tras ser invocada se despliega en el entorno un marco de interacción con los controles necesarios para ello y se puebla en base a su modelo descriptivo, mostrando la secuencia de *tareas* constituyentes.

Tareas *MDDE* (*mTask*). En el *MDDE framework*, el elemento operativo básico, representativo de alguna operación útil para el desarrollo de sistemas o para la gestión del propio entorno, se denomina *mTask*. Así, las *mTasks* son las actividades elementales que constituyen las *mTools* del entorno y se introducen con varios objetivos: i) Posibilitar la reutilización de operaciones compartidas por diferentes *mTools*; ii) ofrecer una interfaz de gestión homogénea que facilite la composición de actividades en los procesos y la interacción con los recursos del entorno y iii) constituir un adaptador para artefactos desarrollados con independencia de él (ver apartado siguiente).

Según su ciclo de ejecución, una *mTask* puede ser *atómica* o *interactiva*. Atómica significa que su ejecución se realiza únicamente en base a datos de configuración iniciales, por lo que se ejecuta sin requerir intervención del operador. En cambio, interactiva significa que sí requiere su intervención, tanto para gestionar información como para decidir las opciones de flujo de control. Por otro lado, según la forma en que las *mTasks* implementan su funcionalidad, pueden clasificarse como *nativas*, esto es, definidas en el entorno y que sólo requieren su invocación directa, o con funcionalidad definida mediante un modelo. Además, también se consideran *mTasks* de tipo adaptador de artefacto (ver apartado siguiente).

Artefactos *MDDE* (*mGadgets*). Son recursos software utilizados por las *mTools* pero que han sido desarrollados fuera del entorno, con independencia de su modelo de referencia. Normalmente, un *mGadget* proporciona una funcionalidad de procesamiento de modelos, presentación de información o de interacción con el operador y tanto su empleo por una *mTool* como su interacción con los recursos del entorno no se

realizan de manera directa, sino que se precisa de un adaptador que permita su invocación, configuración, manifestación de estatus e interacción con el operador.

Un *mGadget* puede ser interno (se ejecuta en el espacio de memoria del propio entorno en que se invoca) o externo (tiene que ejecutarse en un espacio de memoria distinto, bien en el mismo procesador que el entorno o en otro diferente – *mGadget* externo y remoto). La gestión de los de tipo externo y su interacción con el entorno se realizan a través de mecanismos de comunicación establecidos en el modelo de referencia de *MDDE* y proporcionados por la plataforma en que se ejecuta el entorno. Asimismo, para su invocación se requiere que en el espacio de memoria en que se ejecuta haya una aplicación *mGadgetLauncher* de lanzamiento de *mGadgets*.

La distinción internos / externos implica una distinción paralela entre las *mTasks* de tipo adaptador. Las de adaptación entre el entorno y uno interno proporcionan los modelos de entrada requeridos por el *mGadget* en el formato adecuado, almacenan en el entorno los modelos de resultados que éste genera y traducen los mensajes de control y estatus entre él y el entorno. Las de adaptación de uno externo requieren la intervención del servicio de comunicación de la plataforma así como la serialización de los modelos y mensajes intercambiados entre ésta y el *mGadget*.

Información no formalizada como modelos. El ámbito externo a un entorno *MDDE* es heterogéneo, por lo que ni se puede ni conviene mantener que la información ha de estar formulada como modelos conformes a metamodelos conocidos desde el entorno. Por ello, en *MDDE* se ha considerado dar soporte a la formulación de información mediante lenguajes textuales, posibilitando así su intercambio con otros entornos o sistemas que tienen su propio lenguaje específico. Con ello también se consigue un mecanismo sencillo para que colaboradores sin acceso al entorno aporten información o la reciban de él y además proporciona un medio perdurable si se prevé que su duración en el tiempo va a ser más prolongada que la supervivencia del propio entorno. Por ejemplo, puede contemplarse la importación o exportación de información textual de tipo XML formalizada a través de plantillas W3C-Schema o lenguajes específicos desarrollados mediante Xtext [3].

MDDE workbench. GUI de referencia que ha de implementar todo entorno *MDDE* para posibilitar la actuación del operador con la información contenida en el espacio de trabajo y con los procesos en el entorno. La **Fig. 1** lo esquematiza. Básicamente se compone de un área superior (secciones S1 y S2), una zona central (sección S3) y una región periférica (secciones S4, S5, S6 y S7). Cada una de estas últimas está formada por uno o más *marcos de interacción*, áreas gráficas con visores y controles complementarios que implementan un mecanismo de interacción con el operador. La *sección de gestión de elementos* (S4) presenta dos marcos de tipo explorador para organizar y acceder a la información (modelos, metamodelos, etc.) en el entorno. La *sección de información sobre elementos* (S5) presenta dos marcos para exponer información del elemento seleccionado en los marcos de S4. La *sección de información de la ejecución de herramientas* (S6) presenta dos marcos para exponer información propia de la ejecución de las *mTools*. Por último, la *sección de gestión de herramientas* (S7) proporciona al operador la capacidad de controlar la ejecución de una *mTool* previamente invocada, permitiéndole establecer su configuración, llevar a cabo de forma controlada su ejecución, supervisar el estatus de ejecución y finalizar o abortar la ejecución.

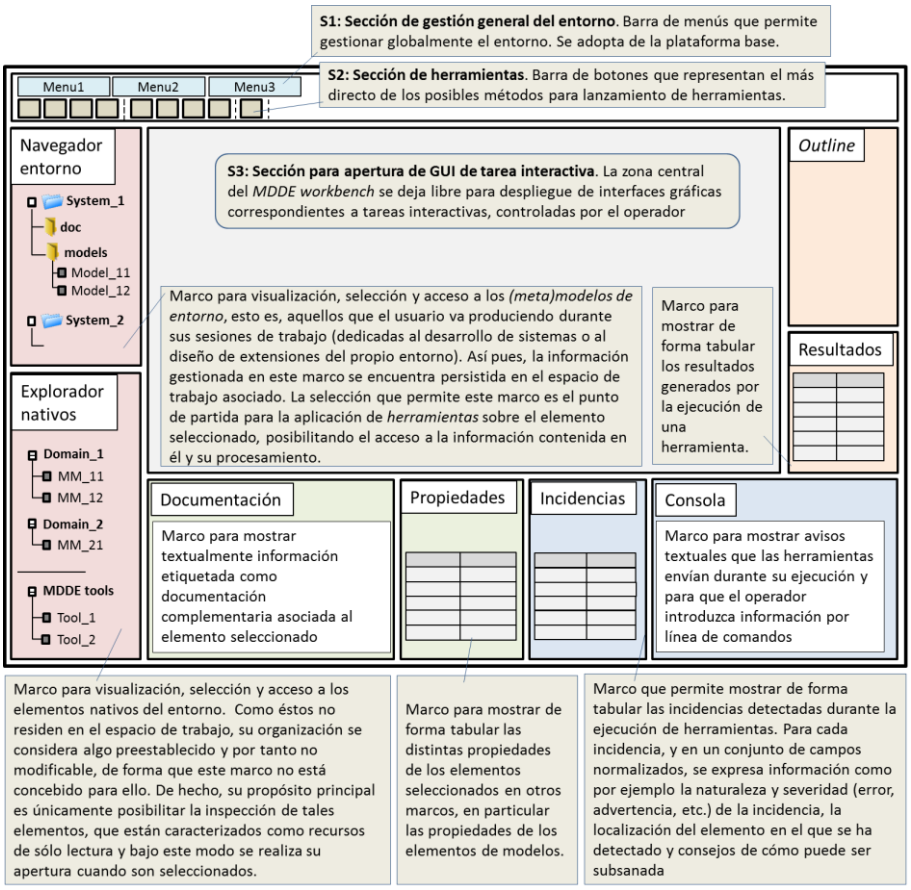

Fig. 1. Esquema del *MDDE workbench.*

La **Fig. 2** muestra en detalle el marco *Tool Outline*, cuya función es mostrar la constitución de una *mTool* (secuencia de *mTasks*) y controlar su ejecución.

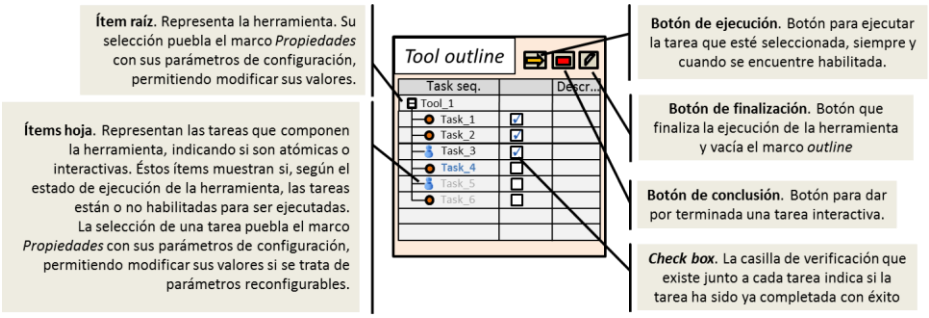

Fig. 2. Elementos de interacción del marco *Tool outline*

3.2 Metamodelado de la parte conceptual del *MDDE framework*

El metamodelo MDDE formaliza los aspectos estructurales del *MDDE framework*. Presenta un diseño que permite especificar un entorno *MDDE* mediante la formulación conjunta de varios modelos conformes a él, tal y como muestra la **Fig. 3**. Se trata de los modelos de formulación de las *mTools* en el entorno, que a su vez encapsulan el modelado de las *mTasks* constituyentes, los modelos de formulación de los *mTaskTypes* definidos en el entorno y los modelos del entorno, que representan propiamente un entorno pero su papel se reduce simplemente a ser un aglutinador de los modelos anteriores. Por *mTaskType* se entiende un ente parametrizado (configurable) cuya realización específica (asignación de valores a sus parámetros de configuración) da lugar a una *mTask* concreta. Así, las *mTasks* contenidas en los modelos de *mTools* son realizaciones de los *mTaskTypes* formulados mediante estos modelos.

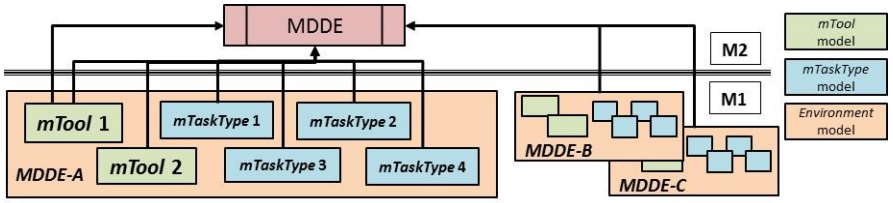

Fig. 3. Especificación de entornos *MDDE* mediante modelos

La **Fig. 4** muestra el núcleo del metamodelo *MDDE*. En función de todo lo expuesto anteriormente, la semántica de las clases mostradas es evidente. `TaskDescriptor` representa el concepto de *mTaskType*. `Task` representa el concepto de *mTask*, como realización concreta de una instancia *mTaskType* apuntada mediante la referencia `descriptor`. `Tool` representa el concepto de *mTool*. Su asociación `tasks` permite que una instancia suya albergue la secuencia de *mTasks* constituyentes, instancias de `Task`. Finalmente, `EnvironmentModel` representa el concepto de entorno. Una instancia suya referencia a través de las asociaciones `tools` y `taskDescriptors` a los conjuntos de *mTools* y *mTaskTypes* que forman la especificación de un entorno *MDDE*.

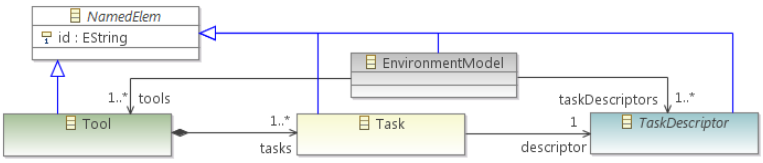

Fig. 4. Núcleo del metamodelo MDDE

Aunque pueda parecer que el metamodelo presenta una estructura convencional, con la clase `EnvironmentModel` como clase contenedor principal, sus asociaciones `tools` y `taskDescriptors` no exhiben carácter de composición. Esta clase simplemente da soporte al modelo aglutinador del resto de modelos. Así, `Tool` y `TaskDescriptor` también poseen rol de contenedor principal y cada modelo de *mTool* o de *mTaskType* cuenta respectivamente con una instancia suya en su raíz.

A continuación, la **Fig. 5** y la **Fig. 6** amplían la exposición del metamodelo MDDE, omitiendo aquí, por razones de espacio, la definición detallada de cada clase mostrada. La especificación completa del metamodelo, junto a su formulación Ecore, pueden encontrarse en http://www.istr.unican.es/members/cesarcuevas/phd/es/mdde.html.

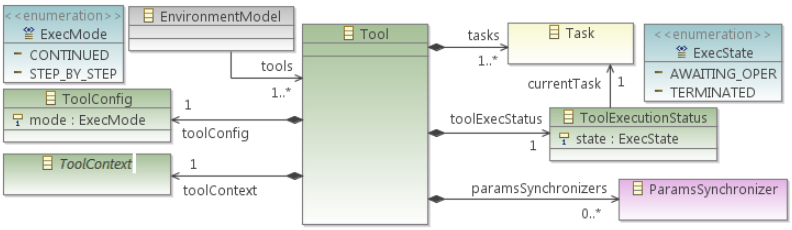

Fig. 5. Clase `Tool`

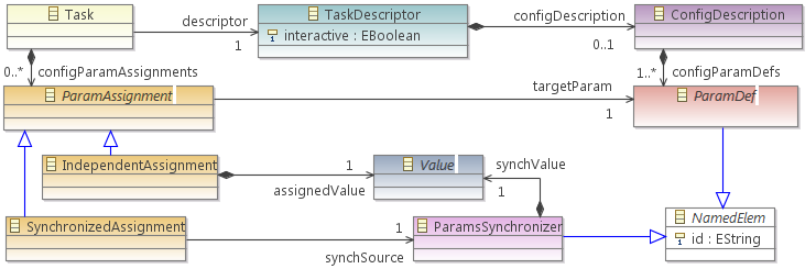

Fig. 6. Definición y asignación de parámetros

4 Ejemplo: Entorno *MDDE-MinimalMAST2*

Como ejemplo de entorno *MDDE* se propone *MDDE-MinimalMAST2*, concebido para dar soporte al análisis y diseño de STREs mediante la metodología MAST. Por motivos de espacio, no se presenta al completo. Su especificación completa junto a la formulación Ecore de los modelos mediante los que se representa así como su implementación sobre Eclipse (*MDDE-Eclipse-MinMAST2*) puede encontrarse en http://www.istr.unican.es/members/cesarcuevas/phd/es/mdde.html.

4.1 Visión general de MAST

MAST (*Modeling and Analysis Suite for Real-Time Applications*) es un entorno para diseño y análisis de STREs, desarrollado por el Grupo ISTR de la Universidad de Cantabria. Está básicamente constituido por una terna de modelos conceptuales (metamodelos MAST – actualmente completando su evolución a MAST-2 [4] –, MAST *Results* y MAST *Traces*) y un conjunto de herramientas que operan sobre modelos conformes.

4.2 Procesos del entorno *MDDE-MinimalMAST2*

El entorno ofrece tres procesos o *mTools*: i) Creación del modelo MAST-2 de un STRE; ii) análisis de planificabilidad y iii) simulación del comportamiento temporal. Dado un STRE modelado mediante MAST-2, la segunda permite aplicar diversos tipos de análisis de planificabilidad empleando las herramientas MAST-1.X, mientras que la tercera permite simular su evolución temporal empleando el simulador *JSimMAST2* [5], produciendo como salida un modelo MAST-2 *Results* y, opcionalmente, un modelo MAST-2 *Traces*. A continuación se expone en detalle esta tercera *mTool*.

La **Fig. 7** esquematiza su constitución en base a su secuencia de *mTasks*. Previamente a ejecutar la simulación propiamente dicha, el proceso contempla verificar la corrección del modelo MAST-2 de entrada (*mTask1*) y su compatibilidad con *JSimMAST2* (*mTask2*). La primera incluye la satisfacción de las restricciones de integridad del metamodelo MAST-2 y la segunda las específicas del simulador. A continuación, es necesario elegir un perfil para la ejecución de la simulación y en base a él transformar (*mTask3*) el modelo de entrada a uno de simulación (conforme al metamodelo SimMAST2). Llegado a este punto, el proceso consiste propiamente en ejecutar y controlar la simulación (*mTask4*), según el mismo perfil de simulación ya establecido.

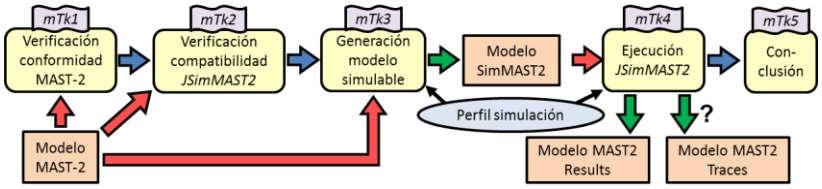

Fig. 7. *mTasks* constituyentes de la *mTool* de Simulación.

Como puede observarse, las *mTasks* expuestas se encuentran ligadas entre sí, pues el modelo de entrada a la *mTask1* lo es también a las *mTask2* y *mTask3*, el modelo de salida de la *mTask3* es modelo de entrada a la *mTask4* y el perfil de simulación establecido para la *mTask3* ha de coincidir con el establecido para la *mTask4*, pues según cuál vaya a ser éste, la generación del modelo simulable sigue unas directrices u otras.

4.3 Tipos de tarea

Para formalizar la constitución de las *mTools*, el entorno define cinco *mTaskTypes*: i) Verificación M2M de restricciones [6]; ii) Transformación MAST-2 → SimMAST2; iii) ejecución de simulación (lanzamiento del *JSimMAST2*); iv) ejecución de análisis de planificabilidad (lanzamiento de la herramienta externa MAST) y v) conclusión de proceso. A continuación se expone en detalle el primero.

M2M verification. Chequeo basado en M2M de si un modelo cumple un conjunto de restricciones especificadas sobre su metamodelo, ofreciendo como resultado un modelo de diagnóstico [6]. Las *mTasks* de este tipo son atómicas y están caracterizadas por poseer los siguientes parámetros de configuración (**Fig. 8**): localización del modelo a verificar, localización donde generar el modelo diagnóstico y el paquete de restricciones frente al que realizar la verificación.

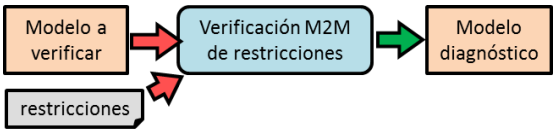

Fig. 8. *mTaskType* Verificación M2M de restricciones

Este *mTaskType* es un ejemplo paradigmático de reutilización, pues *mTasks* de este tipo pueden formar parte de multitud de *mTools* en las que antes de procesar modelos sea necesario realizar las verificaciones oportunas en cuanto a cumplimiento de restricciones. Es lo que sucede precisamente en las *mTools* de este entorno, pues mediante este *mTaskType* se da soporte, por ejemplo, a las *mTask1* y *mTask2* de la **Fig. 7**.

5 Herramientas y recursos para diseño de los entornos

MDDE define una variante del *workbench* orientada a los diseñadores de entornos específicos. Se llama *extension workbench* y su *layout* simplifica el de la versión original, prescindiendo de los marcos *Outline*, *Resultados* y *Consola*. Además, los contenidos de los marcos *Explorador de entorno* y *Navegador de elementos nativos* se ven modificados (**Fig. 9**), ya que ahora el ámbito de desarrollo es el propio dominio *MDDE* y al usuario le resultan de interés los *mTaskTypes* definidos en el entorno, de los cuales puede incluir realizaciones en las *mTools* que diseñe. Asimismo, también puede diseñar nuevos *mTaskTypes*. En el *Navegador de entorno* se ocultan cualesquiera elementos propios de sesiones de trabajo relativas al diseño de sistemas y en su lugar se muestra una sencilla estructura de contenedores en la que almacenar los modelos de *mTools* y *mTaskTypes* que el diseñador de entornos formule. En cuanto al *Explorador de elementos nativos*, se ocultan cualesquiera elementos propios del ámbito específico de desarrollo de sistemas y en su lugar se muestra el metamodelo MDDE y los *mTaskTypes* definidos en el entorno. Además, ahora el marco posee controles para añadir nuevas *mTools*, nuevos *mTaskTypes* o refrescar la visualización.

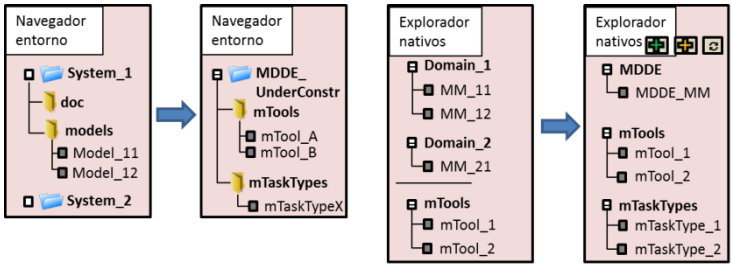

Fig. 9. Marcos *Explorador de entorno* y *Navegador de elementos nativos*

Creación de nuevas *mTools* y/o *mTaskTypes*. Se proporcionan los asistentes *NewTool* y *NewTaskType*, invocables respectivamente con los botones *Add mTool* y *Add mTaskType* del *Explorador de elementos nativos*. Su funcionamiento puede resumirse en que, mediante el cuadro de diálogo que emerge con la invocación, se introduce el

nombre de la *mTool* / *mTaskType* a desarrollar y tras ello se inicializa en memoria un modelo de *mTool* / *mTaskType* con la instancia `Tool` / `TaskDescriptor` contenedor principal adecuadamente inicializada. En el caso de *mTool*, la instancia `Tool` contiene dos instancias `Task`. La primera, *mTask_1*, no se encuentra asignada a ningún descriptor, mientras que la segunda, identificada *Termination*, es de tipo finalización de proceso. A continuación, el modelo generado se persiste en un fichero `*.tool.mdde` / `*.tasktype.mdde` con el nombre introducido por el usuario y ruta `workspace/MDDE_UnderConstr/tools | taskTypes`, donde `workspace` es la ruta del espacio de trabajo y la estructura de contenedores `MDDE_UnderConstr/tools | /taskTypes` se crea si no existe. Por último, se abre el modelo en el área de edición para que el operador complete su construcción.

Registro de las extensiones desarrolladas. Puesto que completar los modelos de *mTools* y *mTaskTypes* puede prolongarse un número indefinido de sesiones de trabajo, éstos permanecen persistidos en el espacio de trabajo, sin ser consideradas extensiones *MDDE* incorporadas de facto al entorno. Cuando se considera completada la formulación de un modelo, ha de procederse a su incorporación al entorno. Para ello, se definen las funcionalidades *Register mTool* y *Register mTaskType*, accesibles respectivamente mediante selección contextual de un modelo de *mTool* o *mTaskType* en el *Navegador de entorno*. El registro implica una fase preliminar de verificaciones (conformidad respecto al metamodelo MDDE y cumplimiento de restricciones) que, en caso de ser superada, conduce al registro propiamente dicho. Esto significa el traslado del modelo desde su ubicación provisional en el espacio de trabajo a la ubicación establecida para las extensiones *MDDE*, desapareciendo del *Navegador de entorno* y apareciendo en el *Explorador de elementos nativos*. A partir de entonces, en el caso de una *mTool*, ya se encuentra disponible para un diseñador de sistemas o, en el caso de un *mTaskType*, para su uso por el propio diseñador de entornos.

6 Trabajo relacionado

Un trabajo próximo al aquí expuesto es la metodología presentada en [7]. En ella se encuentran paralelismos con nuestro trabajo tanto en el objetivo final de facilitar el uso de MDSE como en la propia metodología en sí, basada en la ejecución semiautomática de flujos de actividades (*workflows*) [8]. Sin embargo, el trabajo se centra principalmente en los usuarios finales de entornos MDSE, tanto ingenieros de lenguajes como modeladores específicos de dominio, buscando agilizar sus actividades típicas. Nuestro trabajo, en cambio, está más orientado hacia la adopción de MDSE por parte de los ingenieros software a la hora de diseñar los entornos de desarrollo específicos de dominio que van a dar soporte a las metodologías propuestas por ellos, que típicamente no son expertos en tecnologías MDSE.

Los autores de [7] parten de una premisa básica: El éxito de MDSE depende tanto de que las herramientas aborden convenientemente las correspondientes tareas como de que permitan aumentar la productividad de los modeladores en sus actividades cotidianas. Sin embargo, la gran multitud de *frameworks* y herramientas de modelado exis-

tentes actualmente, aunque con funcionalidades basadas en fundamentos comunes, supone un importante hándicap para los usuarios. Éstos necesitan adaptarse a cada herramienta, pues cada una impone su propio proceso de desarrollo, el cual difiere ampliamente según la herramienta usada. Así, el objetivo planteado es incrementar la productividad de los modeladores en sus actividades habituales mediante la automatización de las tareas comunes realizadas con las herramientas MDSE actuales, desde operaciones simples (apertura, cierre o salvado de modelos) a tareas más complejas, como generación de artefactos para un DSL. También se considera la integración de tareas manuales. Para ello proponen una solución dirigida por modelos en la que se definen *workflows* reusables en forma de plantillas parametrizadas. Mediante la parametrización se consigue un mecanismo de reutilización para minimizar el número de *workflows* a ser creados, en base a que diversidad de tareas ocurren repetidamente en diferentes *workflows*. Puesto que la solución propuesta sigue el paradigma *model-driven*, la ejecución de *workflows* se modela completamente en forma de transformaciones de modelos, haciendo que sea reusable y portable en varias herramientas MDSE. Como parte de la metodología, se propone un DSL inspirado en los diagramas de actividad UML para el diseño de tales plantillas de *workflows*.

La metodología propuesta ha sido implementada en su propia herramienta AToMPM [9], aunque ha de poder ser implementada sobre diversidad de *frameworks* base, tanto soportando metamodelado a dos niveles como metamodelado profundo (*deep metamodelling*) [10, 11].

7 Conclusiones y trabajo futuro

El diseño de un entorno de desarrollo basado en MDSE no sólo requiere diseñar metamodelos que formalicen el soporte de la información y herramientas que lleven a cabo las transformaciones de ésta, sino también diseñar procesos que engloban conjuntos de modelos generados encadenando y/o iterando la aplicación de herramientas en el entorno bajo la supervisión del operador. Aunque concebir estos procesos sea responsabilidad del diseñador de entornos, que planea las estrategias con las que utilizar el entorno, su implementación en base a la infraestructura MDSE proporcionada por la plataforma de soporte al entorno queda, por su complejidad, fuera de su capacidad y conocimiento.

Para facilitar la tarea de este agente, se propone *MDDE*, una concepción genérica de entornos basados en MDSE que incluye la definición de un modelo de referencia para el diseño de entornos y de un conjunto de recursos de soporte que facilitan al experto diseñador de entornos su especificación e implementación. Siempre que éste acepte el modelo de referencia, los procesos se formulan como modelos que describen los modelos y herramientas participantes y las interacciones requeridas con el operador. Estos modelos descriptivos de procesos son interpretados por una herramienta ofrecida por el entorno, permitiendo su ejecución.

Por el momento, la validación de *MDDE* se encuentra en una etapa inicial, pues sólo se ha abordado una implementación, a modo de prueba de concepto, sobre la plataforma

Eclipse y orientada al ámbito de los STREs haciendo uso de las herramientas del entorno MAST. Es necesario seguir abordando otros ámbitos de desarrollo de sistemas software que permitan identificar puntos de extensión para *MDDE*, ampliándola en consecuencia y realizando las correspondientes implementaciones.

Agradecimientos. Este trabajo ha sido financiado parcialmente por el Gobierno de España con referencia TIN2014-56158-C4-2-P (M2C2).

Referencias bibliográficas

1. M. González Harbour, J. J. Gutiérrez García, J. L. Medina, J. C. Palencia, J. M. Drake, J. M. Rivas, P. López Martínez and C. Cuevas, "MAST: Bringing response-time analysis into real-time systems engineering," in *Workshop on Real-Time Systems: The Past, the Present, and the Future,*Anonymous York (UK): CreateSpace Independent Publishing Platform, 2013, pp. 42-59.
2. C. Cuevas, "Metaherramientas MDE para el diseño de entornos de desarrollo de sistemas distribuidos de tiempo real," 2016.
3. M. Eysholdt and H. Behrens, "Xtext: Implement your language faster than the quick and dirty way," in *Proceedings of the ACM International Conference Companion on Object Oriented Programming Systems Languages and Applications Companion*, 2010, pp. 307-309.
4. C. Cuevas, J. M. Drake, P. López Martínez, J. J. Gutiérrez García, M. González Harbour, J. L. Medina and J. C. Palencia, "MAST 2 Metamodel," 2012.
5. C. Cuevas, P. López Martínez and J. M. Drake, "JSimMAST: Simulador de sistemas de tiempo real diseñados con paradigmas avanzados," in Congreso Español de Informática (CEDI), III Simposio de Sistemas de Tiempo Real, Valencia (Spain), 2010, pp. 69-74.
6. C. Cuevas, P. López Martínez and J. M. Drake, "Model-driven approach for verifying conformity of models in the presence of constraints," in 4th International Conference on Model-Driven Engineering and Software Development (MODELSWARD), Rome (Italy), 2016, pp. 455-466.
7. M. A. Gamboa and E. Syriani, "Automating activities in MDE tools," in 4th International Conference on Model-Driven Engineering and Software Development (MODELSWARD), Rome (Italy), 2016, .
8. N. Russell, A. H. Ter Hofstede and N. Mulyar, "Workflow Controlflow Patterns: A revised view," 2006.
9. E. Syriani, H. Vangheluwe, R. Mannadiar, C. Hansen, S. Van Mierlo and H. Ergin, "AToMPM: A web-based modeling environment." in *Demos/Posters/StudentResearch@ MoDELS*, 2013, pp. 21-25.
10. J. De Lara and E. Guerra, "Deep meta-modelling with MetaDepth," in *Objects, Models, Components, Patterns*Anonymous Springer, 2010, pp. 1-20.
11. A. Rossini, J. de Lara, E. Guerra, A. Rutle and U. Wolter, "A Formalisation of Deep Metamodelling," *Formal Aspects of Computing*, vol. 26, pp. 1115-1152, 2014.